Docket No.: 61282-058 **PATENT**

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of	:	Customer Number: 20277
Masayoshi KODAMA, et al.	:	Confirmation Number:
Serial No.:	:	Group Art Unit:
Filed: January 30, 2004	:	Examiner:

For: MEMORY MANAGING SYSTEM AND TASK CONTROLLER IN MULTITASK SYSTEM

CLAIM OF PRIORITY AND
TRANSMITTAL OF CERTIFIED PRIORITY DOCUMENT

Mail Stop Patent Appication
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

In accordance with the provisions of 35 U.S.C. 119, Applicants hereby claim the priority of:

Japanese Patent Application No. 2003-282887, filed July 30, 2003

A Certified copy is submitted herewith.

Respectfully submitted,

MCDERMOTT, WILL & EMERY

Michael E. Fogarty
Registration No. 36,139

600 13th Street, N.W.
Washington, DC 20005-3096
(202) 756-8000 MEF:prg
Facsimile: (202) 756-8087
Date: January 30, 2004

61282-058
Kodama et al.
January 30, 2004
McDermott, Will & Emery

日本国特許庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出願年月日
Date of Application: ２００３年　７月３０日

出願番号
Application Number: 特願２００３－２８２８８７

[ST. 10/C]: [ＪＰ２００３－２８２８８７]

出願人
Applicant(s): 松下電器産業株式会社

２００３年１０月１７日

特許庁長官
Commissioner,
Japan Patent Office 今井康夫

出証番号　出証特２００３－３０８５７３２

【書類名】　特許願
【整理番号】　5037750003
【提出日】　平成15年 7月30日
【あて先】　特許庁長官殿
【国際特許分類】　G06F 13/10
【発明者】
　【住所又は居所】　大阪府門真市大字門真１００６番地　松下電器産業株式会社内
　【氏名】　小玉　将義
【発明者】
　【住所又は居所】　大阪府門真市大字門真１００６番地　松下電器産業株式会社内
　【氏名】　小林　圭太
【特許出願人】
　【識別番号】　000005821
　【氏名又は名称】　松下電器産業株式会社
【代理人】
　【識別番号】　100105647
　【弁理士】
　【氏名又は名称】　小栗　昌平
　【電話番号】　03-5561-3990
【選任した代理人】
　【識別番号】　100105474
　【弁理士】
　【氏名又は名称】　本多　弘徳
　【電話番号】　03-5561-3990
【選任した代理人】
　【識別番号】　100108589
　【弁理士】
　【氏名又は名称】　市川　利光
　【電話番号】　03-5561-3990
【選任した代理人】
　【識別番号】　100115107
　【弁理士】
　【氏名又は名称】　高松　猛
　【電話番号】　03-5561-3990
【選任した代理人】
　【識別番号】　100090343
　【弁理士】
　【氏名又は名称】　栗宇　百合子
　【電話番号】　03-5561-3990
【手数料の表示】
　【予納台帳番号】　092740
　【納付金額】　21,000円
【提出物件の目録】
　【物件名】　特許請求の範囲　1
　【物件名】　明細書　1
　【物件名】　図面　1
　【物件名】　要約書　1
　【包括委任状番号】　0002926

【書類名】特許請求の範囲
【請求項１】
割込み発生に応じて、ＣＰＵ内情報を被割込みタスクのタスクスタックに格納して第１領域となし、該格納後のスタックポインタの値を割込み処理スタックの所定の第１位置に格納し、スタックポインタを前記割込み処理スタックの所定の第２位置にセットし、割込み処理を開始するマルチタスクシステムのスタック管理方法であって、
　前記割込みスタック処理スタックの先頭アドレスを、特定タスクのタスクスタックにおける前記第１領域内の所定位置に一致させることを特徴とするスタック管理方法。
【請求項２】
前記第１領域は、割込み処理からの復帰制御に必須なタスク制御情報を格納する第２領域と、該第２領域の直後に配置され前記タスク制御情報以外のＣＰＵ内情報を格納する第３領域とからなり、
　前記特定タスクのタスクスタックにおける第１領域内の所定位置は、該第１領域における前記第３領域の先頭アドレスであることを特徴とする請求項１記載のスタック管理方法。
【請求項３】
前記割込み処理スタックの所定の第１位置は、該割込み処理スタックの先頭アドレスであることを特徴とする請求項１記載のスタック管理方法。
【請求項４】
前記割込み処理スタックの所定の第２位置は、前記割込み処理スタックの所定の第１位置の直後に位置することを特徴とする請求項１記載のスタック管理方法。
【請求項５】
前記第１領域は、割込み処理からの復帰制御に必須なタスク制御情報を格納する第２領域と、該第２領域の直後に配置され前記タスク制御情報以外のＣＰＵ内情報を格納する第３領域とからなり、
　前記特定タスクのタスクスタックにおける第１領域内の所定位置は、該第１領域における前記第３領域の先頭アドレスに所定のアドレス差分値を補正したアドレスであることを特徴とする請求項１記載のスタック管理方法。
【請求項６】
前記アドレス差分値は、前記特定タスクにおいて使用されるＣＰＵ内情報を格納するために必要な領域を指定するように与えられるものであることを特徴とする請求項５記載のスタック管理方法。
【請求項７】
前記アドレス差分値は、あらかじめ定数として与えられることを特徴とする請求項５記載のスタック管理方法。
【請求項８】
前記アドレス差分値は、前記特定タスクの実行により設定されるものであることを特徴とする請求項５記載のスタック管理方法。
【請求項９】
前記タスク制御情報は少なくとも、プログラムカウンタと、ＣＰＵの状態を表すプログラムステータス語（ＰＳＷ）とを含むことを特徴とする請求項２記載のスタック管理方法。
【請求項１０】
前記特定タスクは、ＣＰＵ内情報を使用せずに自己ループするアイドルタスクであることを特徴とする請求項１記載のスタック管理方法。
【請求項１１】
前記特定タスクは、低消費電力モードへの移行と復帰を制御するタスクであることを特徴とする請求項１記載のスタック管理方法。
【請求項１２】
前記第２領域を設けず、前記タスク制御情報および所定範囲の前記ＣＰＵ内情報は被割込みタスクのタスクスタックに格納せずにタスクスタックと異なるメモリ領域に格納し、前

記所定範囲のＣＰＵ内情報以外のＣＰＵ内情報は前記第３領域に格納されることを特徴とする請求項２記載のスタック管理方法。

【請求項１３】

割込み発生に応じて、ＣＰＵ内情報を被割込みタスクのタスクスタックに格納して第１領域となし、該第１領域格納後のスタックポインタの値を割込み処理スタックの所定の第１位置に格納し、スタックポインタを前記割込み処理スタックの所定の第２位置にセットし、割込み処理を開始するマルチタスクシステムのスタック制御装置であって、

前記割込み処理スタックの先頭アドレスを、特定タスクのタスクスタックにおける前記第１領域内の所定位置に一致させる制御機構を備えることを特徴とするスタック制御装置。

【請求項１４】

前記第１領域は、割込み処理からの復帰制御に必須なタスク制御情報を格納する第２領域と、該第２領域の直後に配置され前記タスク制御情報以外のＣＰＵ内情報を格納する第３領域とからなり、

前記特定タスクのタスクスタックにおける第１領域内の所定位置は、該第１領域における前記第３領域の先頭アドレスとすることを特徴とする請求項１３記載のスタック制御装置。

【請求項１５】

前記第１領域は、割込み処理からの復帰制御に必須なタスク制御情報を格納する第２領域と、該第２領域の直後に配置され前記タスク制御情報以外のＣＰＵ内情報を格納する第３領域とからなり、

前記特定タスクのタスクスタックにおける第１領域内の所定位置は、該第１領域における前記第３領域の先頭アドレスに所定のアドレス差分値を補正したアドレスとすることを特徴とする請求項１３記載のスタック制御装置。

【請求項１６】

前記アドレス差分値は、前記特定タスクにおいて使用されるＣＰＵ情報を格納するために必要な領域を示すアドレス差分値格納手段に保持されることを特徴とする請求項１３記載のスタック制御装置。

【請求項１７】

前記アドレス差分値は、あらかじめ定数として与えられることを特徴とする請求項１３記載のスタック制御装置。

【請求項１８】

前記アドレス差分値は、前記特定タスクの実行により設定されるものであることを特徴とする請求項１３記載のスタック制御装置。

【請求項１９】

請求項５から８のうちいずれか１項記載のスタック管理方法、または請求項１５から１８のうちいずれか１項記載のスタック制御装置における前記アドレス差分値を自動的に生成するコンパイラ。

【書類名】明細書
【発明の名称】マルチタスクシステムにおけるメモリ管理方式およびタスク制御装置
【技術分野】
【０００１】
本発明は、ソフトウェアのスタックメモリの管理方法、特にマルチタスクシステムを実行する場合のスタックメモリ使用量を削減するプログラム構造に関する。
【背景技術】
【０００２】
プログラムにおける制御の複雑化の進展により、コンピュータによる仕事の単位であるタスクを一度に２つ以上処理することができるマルチタスクシステムが一般的になってきている。このマルチタスクシステムを用いることによって、複数のタスクを効率よく切替えて実行することができる。
【０００３】
図３は、従来のマルチタスクシステムにおける、ＲＡＭの使用状況の一例を摸式的に示した図である。図３における番号がふられた領域は、各タスクで使用するスタックメモリ領域（以下、スタックという）を示している。すなわち、ＲＡＭ 上にはマルチタスクシステムによって実行される複数のタスク１ 、タスク２、・・・、タスクｎの複数のスタックと、アイドルタスク用スタック、割込み処理専用のスタックとが構築されることとなる。
【０００４】
各スタックは各々、スタック３０１、３１１、３２１、戻りＰＣ保存領域３０２、３１２、３２２、ＰＳＷ保存領域３０３、３１３、３２３及びＣＰＵレジスタ保存領域３１４、３２４、３０４によって構成されている。また、割込み処理用スタックは、割込み発生時のタスクＳＰを保存しておく戻りＳＰ保存領域３０５と、割込み処理シーケンスで使用する割込み処理用スタック領域３０６によって構成される。
【０００５】
ここで、割込み処理とはタスク（通常処理）が実行されているときに、このタスクを一時的に中断して、例えばタイマ等の制御により一定の時間毎に実行される処理や、外部要因によって実行される処理などである。
【０００６】
例えば、タスク１が動作している時に割込みが発生した場合には、スタック３１１を用いて実行されているタスク１は一時的に中断される。このとき、現在のＰＣレジスタの値は戻りＰＣ保存領域３１２に保存され、現在のＰＳＷレジスタの値はＰＳＷ保存領域３１３に保存される。さらに、タスク１で使用されていたＣＰＵレジスタの値をＣＰＵレジスタ保存領域３１４に保存する。次に、タスク１のＳＰレジスタの値を割込み専用スタック領域内の戻りＳＰ保存領域３０５に保存し、ＳＰレジスタの値をＳＰ保存領域３０５と割込みで使用するスタック領域３０６の境界、すなわちスタック領域３０６のボトム領域を指すように設定することで、スタック領域をタスクスタックから割込み処理用スタックへ切替える。
【０００７】
割込み処理が終了するときは、ＳＰ保存領域３０５に保存しておいた値をＳＰレジスタに設定することでタスクスタックに切替える。その後は、ＣＰＵレジスタ保存領域３１４に保存しておいた値を各ＣＰＵレジスタへ設定し、ＰＳＷ保存領域３１３と戻りＰＣ保存領域３１２に保存しておいた値も同様にＰＳＷレジスタとＰＣレジスタに復帰することで、元のタスク１へ復帰することができる。このような構成にすることにより、マルチタスクシステムによって複数のタスクを実行中に所定の割込み処理を行なうことができるようになっている（例えば、特許文献１参照）。
【特許文献１】特開平０８－１２３６９８号公報
【発明の開示】
【発明が解決しようとする課題】

【０００８】
しかしながら、図３に示したスタック構造では、割込み発生時に保存するＣＰＵレジスタの退避容量が、タスクの処理内容に関わらず一意に決まったものであり、この中には保存する必要のないレジスタも多く含まれている可能性があり、不必要にメモリを消費している。これに対しては、例えば割込み処理の先頭部分で、保存すべきＣＰＵレジスタの種類を判別するような方法があるが、割込み応答性能を悪化させてしまうという問題点を有している。

【課題を解決するための手段】
【０００９】
本発明は、このような問題点を解決するもので、マルチタスクシステムを実行するｎ個のタスクスタックと、前記ｎ個の各タスクスタックに対して共有されて機能する割込み処理用スタックと、割込み処理が発生したときに実行中のタスクスタックに退避されるＰＣ、ＰＳＷ、ＣＰＵレジスタとがあり、前記割込み処理用スタック領域は前記ｎ個のタスクスタックのいずれか１つのタスクスタックと共有して使用される。上記割込み処理が発生したときに現在のタスクスタックにＰＣ、ＰＳＷ、ＣＰＵレジスタの値を退避したのちにスタックポインタを割込み処理側に切替え、上記割込み処理が終了したときにはスタックポインタをタスクスタックに切替えた後に、上記タスクスタック上に保存されたＰＣ、ＰＳＷ、ＣＰＵレジスタの値をタスクスタックから復帰させ、タスクの動作を再開させるようにする。

【発明の効果】
【００１０】
以上の説明により、この発明の効果として、リアルタイムＯＳを用いたシステムのタスクスタック領域と割込みスタック領域とを重複して使用することで、システム全体でのメモリ使用量を削減できるということが言える。したがって、小容量のメモリでマルチタスクシステムを稼動させることができるという利点がある。

【発明を実施するための最良の形態】
【００１１】
以下に、本発明の実施の形態について、図面を参照しながら説明する。
（実施の形態１）
図１は、本発明のマルチタスクシステムにおけるスタックの使われ方を示す図である。このマルチタスクシステムは、図２に示す従来のマルチタスクシステムに比べて割込み処理用スタック１０７の配置場所に特徴がある。
【００１２】
図２に示す従来のマルチタスクシステムは、３つのタスク１０２、１０４、１０６と１つの割込み処理プログラム１０８がオペレーティングシステム（以下ＯＳという）１０９上で動作しており、それぞれのタスクやプログラムには独立にスタック領域１０１、１０３、２０５、２０７が用意されている。また、３つのタスクのうち、最も優先度の低いタスクをアイドルタスク１０６と呼び、このアイドルタスクが動作しているときは、システムが処理すべきタスク処理がない状態であり、外部からの割込み要求が入ってくるのを待っている状態を示している。
【００１３】
一方、図１に示す本発明のマルチタスクシステムも同様に、３つのタスク１０２、１０４、１０６と１つの割込み処理プログラム１０８がＯＳ１０９の上で動作しており、それぞれのタスクやプログラムには独立にスタック領域１０１、１０３、１０５、１０７が用意されている。また、３つのタスクのうち、最も優先度の低いアイドルタスク１０６も同様に備えている。ただし、割込み処理用のスタック１０７は、アイドルタスク用スタック領域１０５に対して重なるように配置されており、割込み処理時のスタック領域はアイドルタスク処理で使用されたスタック領域を上書き（スタック破壊）しながら動作する。
【００１４】
図３は、従来のマルチタスクシステムにおいて、通常のタスクスタック領域が２つ、ア

イドルタスク用のスタック領域が１つ、割込み処理用スタック領域が１つある場合のスタックの内容を示す図である。

【００１５】

図３において、まずタスク１が動作しているときは、スタックポインタ（以下ＳＰ）がタスク１用のスタック領域３１１を指しており、タスク１はこの領域を使用しながらプログラムを処理している。この状態で割込みが発生し、ＣＰＵがその割込みを受理すると、ハードウェア的にプログラムカウンタ（以下、ＰＣ）とプロセッサステータスワード（以下、ＰＳＷ）の内容をそれぞれスタック領域３１２、３１３に保存する。ＳＰの値は、自動的にＰＣとＰＳＷのサイズ分だけ減算される。次に、ＣＰＵはＰＣの値を割込み処理プログラムの先頭位置へ移動させ、ＯＳ割込み入口処理が実行される。

【００１６】

図５は、従来のマルチタスクシステムにおける、タスク動作中に割込みを受理したときのＯＳ割込み入口処理のフローチャートである。以下では、タスク１の動作中に割込みが発生した場合について説明する。まず、ステップＳ５０１でＣＰＵレジスタの値をスタック領域３１４へ保存する。ＣＰＵレジスタの数は、ＣＰＵの種類によって異なるが、最も簡単な実装はＣＰＵが搭載している全てのＣＰＵレジスタを保存することである。次に、ステップＳ５０２で、現在のＳＰの値（これは該当するタスクスタックの最終アドレスと等しい）を割込み処理用スタック領域のボトム位置にある戻りＳＰ保存領域３０５に保存する。これは、割込み処理終了時にＳＰの指す場所を再びタスクスタックへ復帰させるために必要な処理である。次に、ステップＳ５０３では、割込み処理用スタック領域３０６のボトムアドレスをＳＰに代入して、スタックをタスクスタック３１４から割込み処理専用スタック３０６へ切替える。このとき、タスクへの戻りＳＰ保存領域３０５を上書きしないように、ＳＰの値を設定することに注意する。最後に、ステップＳ５０４にて、アプリケーションによって定義された割込みハンドラ関数をコールし、ステップＳ５０５にてアプリケーションの割込みハンドラプログラムを実行する。ステップＳ５０５からは。関数リターンという形でＯＳ割込み入口処理へ復帰し、その後はＯＳ割込み出口処理へと移行する。

【００１７】

図８は、割込み処理を終了しタスクへ復帰するときのＯＳ割込み出口処理のフローチャートである。ＯＳ割込み出口処理では、まず、ステップＳ８０１でタスクへの戻りＳＰ保存領域３０５に保存されている値をＳＰへ復帰させる。これにより、割込み処理用スタック領域３０６は未使用状態になる。一般的なＯＳでは、ステップＳ８０２にて遅延ディスパッチ処理を行なう。これは、割込み処理からタスクへ復帰する際に、タスクスイッチを行なうべきかどうかを判定し、タスクスイッチを行なう必要があれば、ＳＰの値を現在のタスク１から他タスク（例えば、タスク２）の適当なスタック番地を指すように変更することで行なう。ただし、本発明では遅延ディスパッチの処理内容に関しては、本発明の主題とするところではないので、特に言及しないこととする。ステップＳ８０３では、ステップＳ５０１で保存しておいたＣＰＵレジスタの値をスタック領域３１４から各ＣＰＵレジスタに復帰し、最後にステップＳ８０４にて割込みによって中断されていたタスク処理に復帰する。このとき、ＣＰＵが用意している割込み処理から復帰する命令を実行することで、ＰＳＷとＰＣの内容をスタックから復帰する。

【００１８】

タスク動作中に割込みを受理し、ＯＳ割込み入口処理から、出口処理を経由してタスクへ復帰するステップは、上記で説明したタスク１でなくても、図３で示したタスク２または、アイドルタスクでも全く同様である。つまり、図３においてタスク１が使用している３１１～３１４のスタック領域を、そのまま３２１～３２４、または３０１～３０４のスタック領域として読み替えればよい。

【００１９】

図４は、本発明のマルチタスクシステムにおいて、通常のタスクスタック領域が２つ、アイドルタスク用のスタック領域が１つある場合のスタックの中身を示す図である。

【００２０】
図４において、まずタスク１が動作しているときはＳＰが３１１に示すタスク１用のスタック領域を指しており、タスク１はこの領域を使用しながらプログラムを処理している。この状態で割込みが発生し、ＣＰＵがその割込みを受理すると、ハードウェア的にＰＣとＰＳＷの内容をそれぞれスタック領域３１２、３１３に保存する。このとき、ＳＰの値は、自動的にＰＣとＰＳＷのサイズ分だけ減算される。次に、ＣＰＵはＰＣの値を割込み処理プログラムの先頭位置へ移動させ、ＯＳ割込み入口処理が実行される。
【００２１】
図６は、タスク動作中に割込みを受理したときの割込み入口処理のフローチャートを示している。ＯＳ割込み入口処理では、まずステップＳ５０１で、ＣＰＵレジスタの値をスタック領域３１４へ保存する。ＣＰＵレジスタの数はＣＰＵの種類によって異なるが、最も簡単な実装はＣＰＵが搭載している全てのＣＰＵレジスタを保存することである。次に、ステップＳ６０２で、現在のＳＰの値（これは該当するタスクスタックの最終アドレスと等しい）を割込み処理用スタック領域のボトム位置にある戻りＳＰ保存領域４０５に保存する。これは、割込み処理終了時にＳＰの指す場所を再びタスクスタックへ復帰させるために必要な処理である。ここで、注意を要するのは、図５における従来のマルチタスクシステムと異なり、戻りＳＰ保存領域４０５が、アイドルタスクのＣＰＵレジスタ保存領域４０４に重なって配置されていることである。次に、ステップＳ６０３では、割込み処理用スタック領域４０６のボトムアドレスをＳＰに代入して、スタックをタスクスタックから割込み処理専用スタックへ切替える。このとき、タスクへの戻りＳＰ保存領域４０５を上書きしないように注意する。最後に、ステップＳ５０４にて、アプリケーションによって定義された割込みハンドラ関数をコールし、ステップＳ５０５にてアプリケーションの割込みハンドラプログラムを実行する。ステップＳ５０５からは関数リターンという形でＯＳ割込み出口処理へと移行する。ＯＳ割込み出口処理については、図８により説明した従来技術と同じ処理となるので、詳細の説明は省略する。
【００２２】
ここで、タスク動作中に割込みを受理し、ＯＳ割込み入口処理からＯＳ割込み出口処理を経由してタスクへ復帰するステップは、上記で説明したタスク１でなくても、図４で示したタスク２についても全く同様である。つまり、図４においてタスク１が使用しているスタック領域３１１～３１４を、そのままスタック領域３２１～３２４として読み替えればよい。
【００２３】
ただし、アイドルタスクが使用しているスタック領域に関しては事情が異なる。従来のマルチタスクシステムでは、アイドルタスクが使用しているスタック領域３０１～３０４は、前記タスク１のスタック領域３１１～３１４にそのまま置き換えることができるが、本発明の場合は、図４に示すスタック領域４０４と割込み処理用スタック領域４０５および４０６とが重なって配置されているので、注意を要する。以下では、アイドルタスクが動作している時に割込みを受理した場合の動作を詳細に説明する。
【００２４】
まず、アイドルタスクが動作しているときはＳＰがアイドルタスクスタック領域４０１内の番地を指しており、アイドルタスクはこの領域を使用しながらプログラムを処理している（ただし、この領域は必ず存在するとは限らず、ゼロバイトの場合もある）。この状態で割込みが発生し、ＣＰＵがその割込みを受理すると、ハードウェア的にＰＣとＰＳＷの内容をそれぞれ戻りＰＣ保存領域４０２、ＰＳＷ保存領域４０３に保存する。ここで、ＳＰの値は自動的にＰＣとＰＳＷのサイズ分だけ減算される。すなわち、ＰＣとＰＳＷがそれぞれ４バイトだった場合、スタックは８バイト分がそれらの保存に消費され、ＳＰを８バイト分だけ減算したことに等しい。次に、ＣＰＵはＰＣの値を割込み処理プログラムの先頭番地に移動させ、ここでＯＳ割込み入口処理が実行される。アイドルタスクが動作しているときに割込みを受理した場合のＯＳ割込み入口処理は図６に示すフローチャートに従って動作する。

【００２５】
一方、ＯＳ割込み出口処理は、図８により説明した従来技術と同じ処理となるが、ステップＳ８０３で復帰するＣＰＵレジスタの値は、既に上書きされているため、ＯＳ割込み入口処理時に保存しておいた値ではないことに注意を要する。
【００２６】
図９は、実施の形態１におけるアイドルタスクの動作を表すフローチャートである。実施の形態１におけるアイドルタスクは、図９に示すように、自分自身への無限ループＳ９０１で構成されるよう実装されている。従って、図８におけるステップＳ８０３において、割込み処理内でＰＣとＰＳＷを除く全てのＣＰＵレジスタが不定の値に上書きされても、アイドルタスクのプログラムがＣＰＵレジスタを全く使用せずに動作しているので、割込み復帰後も何ら問題は生じない。
【００２７】
図１２は、上記で説明した本発明のマルチタスクシステムを実現するタスク制御装置を示している。本タスク制御装置は、ＰＣ１２０５が現在実行中のプログラムの命令アドレス１２０９を指しており、この値はバスコントロールユニット（以下ＢＣＵ）１２１２を通して、外部メモリ（ＲＯＭ）１２１３から命令データ１２１０を取得する。命令データ１２１０は、命令実行制御部１２０６でデコード処理され、デコードされた命令の種別に応じて、タスク制御装置１２００全体の動作を決定する。命令データが外部メモリへのデータ書込みを行なう命令であった場合、命令実行制御部１２０６は、ＣＰＵレジスタ１２０１から必要なアドレス値を読出して演算ユニット（以下ＡＬＵ）１２０４に渡し、オペランドアドレス１２０７を、ＢＣＵ１２１２を経由して外部メモリ（ＲＡＭ）１２１４へアドレスデータとして渡す。同時に、ＣＰＵレジスタ１２０１の中からオペランドデータ１２０８を読み出し、ＢＣＵ１２１２を経由して外部メモリ（ＲＡＭ）１２１４へデータを書込む。なお、本タスク制御装置は、割込み制御部１２１１を搭載しており、周辺機能１２１５からの割込み要求を受理して、命令実行制御部１２０６へ通知し、現在実行中のプログラムを中断する機能を備えている。
【００２８】
さらに本タスク制御装置は、割込み処理時のスタックアドレス制御のためにＳＰ差分定数１２５０を備えている。あるタスクの実行中に割り込みが発生した場合は、ＳＰ１２０２の値をオペランドアドレス１２０７およびＢＣＵ１２１２を経由して外部メモリ（ＲＡＭ）１２１４へタスクスタックのアドレスとして渡し、ＰＣ１２０５およびＰＳＷ１２０３の値をタスクスタックに保存した後、その分だけＳＰの値を更新する。続けて、図６のフローチャートに示すように、ＣＰＵレジスタの内容をタスクスタックに保存し、その分だけＳＰの値を更新する。
【００２９】
次に、この最終のＳＰ１２０２の値を割込み処理スタックのＳＰ保存領域４０５に保存する。ＯＳ割込み入口処理プログラムあるいは割込み制御部１２１１には割込み処理スタックの重畳先としてアイドルタスクスタックのＣＰＵレジスタ保存領域４０４の先頭アドレスが予め与えられている。また、ＳＰ差分定数１２５０には、アイドルタスクスタックのＣＰＵレジスタ保存領域４０４の先頭アドレスに対する割込み処理スタックの先頭アドレスのアドレス差分値が与えられている。
【００３０】
割込み処理スタックの重畳先、すなわちアイドルタスクスタックのＣＰＵレジスタ保存領域の先頭アドレスがＯＳ割込み入口処理プログラムあるいは割込み制御部１２１１から与えられ、これとＳＰ差分定数１２５０の値をＡＬＵ１２０４で演算し、これをオペランドアドレス１２０７およびＢＣＵ１２１２を経由して外部メモリ（ＲＡＭ）１２１４へアドレスとして渡すことで、前記最終のＳＰの値を保存する。ここで、本実施の形態ではＳＰ差分定数１２５０に０を与えることで、割込み処理スタックのＳＰ保存領域４０５の先頭アドレスをアイドルタスクスタックのＣＰＵレジスタ保存領域の先頭アドレスに一致させる。この先頭アドレスをＳＰ保存領域分だけ更新してＳＰにセットすることで、割込み

処理スタックが使用可能になり割込み処理が先に進められる。

【００３１】

以上のように、割込み処理スタックをアイドルタスクのタスクスタックに重畳することにより、アイドルタスクのタスクスタックの使用されないＣＰＵレジスタ保存領域を有効活用することができる。

【００３２】

（実施の形態２）

発明の実施の形態１で説明したマルチタスクシステムは、図９に示すような単純な無限ループのみで構成されたアイドルタスク以外にも、図１０のフローチャートに示すような他の処理を含むようなアイドルタスクでも実現することができる。図１０において、アイドルタスクプログラムはステップＳ１００１に示すように、ＣＰＵの動作モードを低消費電力モードへ移行する処理を行なう。ここで、ＣＰＵが低消費電力モードへ移行した後は、割込みが受理されるまでその後の命令は実行されない。ＣＰＵに割込み要求が入り、低消費電力モードが解除されると、ＣＰＵは通常の動作モードに復帰すると同時に、図７のフローチャートに示したＯＳ割込み入口処理を実行することとなる。

【００３３】

ここで、ＣＰＵの動作モードを低消費電力モードへ移行させる際に、ＣＰＵレジスタのうちの１つであるｒ０レジスタを使用するとする。この場合、割込み処理の前後でもｒ０レジスタの値は保存されていなければない。

【００３４】

図１１は、この場合のアイドルスタックと割込み処理用スタックの使用方法を表す図である。まず、アイドルタスクが動作しているときは、ＳＰがアイドルタスク用のスタック領域１１０１を指しており、アイドルタスクはこの領域を使用しながらプログラムを処理している。上記プログラムが、ｒ０レジスタを使用している最中に割込みが発生し、ＣＰＵがその割込みを受理すると、ハードウェア的にＰＣとＰＳＷの内容をそれぞれスタック領域１１０２、１１０３に保存する。ＳＰの値は、自動的にＰＣとＰＳＷのサイズ分だけ減算される。ＣＰＵは、ＰＣの値を割込み処理プログラムの先頭番地へ移動させ、図７に示すＯＳ割込み入口処理が実行される。

【００３５】

図７における、ＯＳ割込み入口処理では、まずステップＳ７０１でＣＰＵレジスタの値をスタック領域１１０７および１１０４へ保存する。この場合、ステップＳ８０１では、必ずｒ０レジスタをＰＳＷ保存領域１１０３のすぐ上にあるｒ０レジスタ保存領域１１０７に保存するようにする。その他のＣＰＵレジスタに関しては保存する順番は問わない。次に、ステップＳ７０２で、現在のＳＰの値を割込み処理用スタック領域のボトム位置１１０５に保存する。実施の形態１と異なり、戻りＳＰ保存領域１１０５は、ｒ０レジスタ保存領域１１０７を上書きしないような位置になっていなければならないが、それ以外のＣＰＵレジスタを保存するＣＰＵレジスタ保存領域１１０４に対しては重なるように配置されている。ステップＳ７０３では、割込み処理用スタック領域１１０６のボトムアドレスをＳＰに代入して、スタックをアイドルタスクスタックから割込み処理用スタック領域１１０６へ切替える。割込み処理用スタック領域１１０６は、先ほどのＣＰＵレジスタ保存領域１１０４と重なっており、先ほど保存したＣＰＵレジスタの値は、実施の形態１の場合と同様に上書きされる。最後に、ステップＳ５０４にて、アプリケーションによって定義された割込みハンドラ関数をコールし、ステップＳ５０５にてアプリケーションの割込みハンドラプログラムを実行する。ステップＳ５０５からは関数リターンという形で、図８で示されるＯＳ割込み出口処理へと移行する。

【００３６】

一方、ＯＳ割込み出口処理は、図８により説明した従来技術と同じ処理となるが、ステップＳ８０３で復帰するＣＰＵレジスタの値は、ｒ０レジスタ保存領域１１０７を除いて、既に上書きされているため、ＯＳ割込み入口処理時に保存しておいた値ではないことに注意を要する。但し、ｒ０レジスタの値は、ｒ０レジスタ保存領域１１０７に保存されて

いた正しい値が復帰される。もともと発生した割込みは、アイドルタスク内でｒ０レジスタを使用中に受理されたものであるが、割込みから復帰した後もアイドルタスクプログラムはｒ０レジスタの内容を使って、ＣＰＵを低消費動作モードへ移行させることが可能である。

【００３７】

以上の説明においては、アイドルタスクの実行中に割り込みが発生した場合のＯＳ割込み入口処理を説明したが、割込み処理スタックがアイドルタスクスタックに重畳されているので、任意のタスクの実行中に割り込みが発生した場合に、ＯＳ割込み入口処理において実施の形態１で説明したものと同様の処理が行われる。すなわち、割込みが発生したタスクのＣＰＵレジスタの内容をタスクスタックに保存した後、最終のＳＰ１２０２の値を割込み処理スタックのＳＰ保存領域１１０５に保存する。

【００３８】

そのために、割込み処理スタックの重畳先、すなわちアイドルタスクスタックのＣＰＵレジスタ保存領域の先頭アドレスがＯＳ割込み入口処理プログラムあるいは割込み制御部１２１１から与えられ、これとＳＰ差分定数１２５０の値をＡＬＵ１２０４で演算し、これをオペランドアドレス１２０７およびＢＣＵ１２１２を経由して外部メモリ（ＲＡＭ）１２１４へアドレスとして渡すことで、前記最終のＳＰの値を保存する。ここで、本実施の形態におけるＳＰ差分定数１２５０には、アドレス差分値として、ｒ０レジスタ１１０７の退避分を考慮した値を与える。その結果、割込み処理スタックのＳＰ保存領域１１０５の先頭アドレスを、アイドルタスクスタックにおけるｒ０レジスタ１１０７の退避領域の次のアドレスにすることができる。

【００３９】

以上のように、割込み処理スタックをアイドルタスクのタスクスタックに重畳することにより、アイドルタスクにおいて一部のＣＰＵレジスタを使用している場合にも、アイドルタスクのタスクスタックの使用されないＣＰＵレジスタ保存領域を有効活用することができる。

【００４０】

（実施の形態３）

実施の形態１および実施の形態２では、アイドルタスクに注目して説明を行なったが、タスクが使用するＣＰＵレジスタが明確に分かっている場合には、アイドルタスク以外の通常のタスクにおいても同じ手法を適用することができる。また、タスクが使用するＣＰＵレジスタが静的に決められない場合でも、上書きしてよいＣＰＵレジスタと上書きしてはならないＣＰＵレジスタを、タスクの処理の中で変数などに情報を残すようにすることで、実施の形態２で説明した手法を同様に適用することができる。さらには、Ｃ言語のような高級言語を使用する場合でも、コンパイラがＣＰＵレジスタに関する情報を、変数もしくはＣＰＵに設けられたＳＰ差分情報格納レジスタに保存するようにすることで、実施の形態２で説明した手法を適用することが可能である。

【００４１】

図１３は、このようなマルチタスクシステムを実現する本発明のタスク制御装置を示している。本発明のタスク制御装置は、実施の形態１および２におけるタスク制御装置に比べて、ＳＰ差分情報格納レジスタ１３００を搭載している点が異なっている。ＳＰ差分情報格納レジスタ１３００の使用方法は任意であるが、ステップＳ５０２で参照するタスク動作時の最終ＳＰ値を保存するアドレス値をコンパイラが予め算出しておき、ＳＰ差分情報格納レジスタ１３００へ保存できる。このようにすれば、任意のタスクの実行中に割込みが受理された場合に、ＯＳ割込み入口処理プログラムあるいは割込み制御部１２１１に予め与えられた、割込み処理スタックの重畳先のタスクスタックにおけるＣＰＵレジスタ保存領域の先頭アドレスとＳＰ差分情報格納レジスタ１３００とにより、ＡＬＵ１２０４で演算した値を、オペランドアドレス１２０７、ＢＣＵ１２１２を経由して外部メモリ（ＲＡＭ）１２１４へアドレスとして渡し、割込みが発生したタスクのＣＰＵレジスタの内容をタスクスタックに保存した後の最終ＳＰ１２０２の値をオペランドデータ１２０８と

して外部メモリ（ＲＡＭ）１２１４へ書出すことが可能である。すなわち、割込み処理スタックのスタート番地をコンパイラなどがＳＰ差分情報格納レジスタ１３００に設定する任意の値で変更することが可能である。もちろん、ＳＰ差分情報格納レジスタ１３００に設定するものは、コンパイラ以外の通常のアプリケーションなどのプログラムであっても構わないし、ハードウェアで設定することも可能である。設定の方法は、本発明の趣旨を越えない部分において、さまざまなものが考えられる。

【００４２】

以上の実施の形態において、ＰＣとＰＳＷの値をタスクスタック上に保存する実施例を説明してきたが、ＣＰＵの種類によってはタスク動作中のレジスタバンクと、割込み処理用のレジスタバンクを別々に搭載しているものもある。このようなＣＰＵでは、割込み受理時にＰＣとＰＳＷの値をタスクスタック上に保存せずに、割込み処理専用のレジスタバンク内にある保存領域にコピーする場合もある。この場合、割込み処理専用のレジスタバンク内に保存されたＰＣとＰＳＷの値を、ソフトウェア的にタスクスタック上に保存するようにすることで、以上の実施の形態と全く同様の効果を奏することが可能である。

【産業上の利用可能性】

【００４３】

本発明のマルチタスクシステムにおけるメモリ管理方式およびタスク制御装置は、リアルタイムＯＳを用いたシステムのタスクスタック領域と割込みスタック領域とを重複して使用することで、システム全体でのメモリ使用量を削減できるということが言える。したがって、小容量のメモリでマルチタスクシステムを稼動させることができるという利点があり、ソフトウェアのスタックメモリの管理方法、特にマルチタスクシステムを実行する場合のスタックメモリ使用量を削減するプログラム構造等として有用である。

【図面の簡単な説明】

【００４４】

【図１】本発明の実施例のスタック構造を模式的に示す図

【図２】従来のマルチタスクシステムにおけるスタック構造を模式的に示す図

【図３】従来のマルチタスクシステムにおけるスタック使用状況を詳細に示す図

【図４】本発明のマルチタスクシステムにおけるスタック使用状況を詳細に示す図

【図５】従来の割込み発生時のスタック切替え処理の内容を示すフローチャート

【図６】本発明の割込み発生時のスタック切替え処理の内容を示すフローチャート

【図７】本発明の低消費動作モードへ移行させる場合の割込み発生時のスタック切替え処理の内容を示すフローチャート

【図８】割込み終了時のスタック切替え処理の内容を示すフローチャート

【図９】空ループの場合の、アイドルタスク処理の内容を示すフローチャート

【図１０】低消費動作モードへ移行する場合の、アイドルタスク処理の内容を示すフローチャート

【図１１】低消費動作モードへ移行させる場合のアイドルタスク用スタックと割込み処理用スタックの構造を詳細に示す図

【図１２】実施の形態１および２におけるタスク制御装置

【図１３】実施の形態３におけるタスク制御装置

【符号の説明】

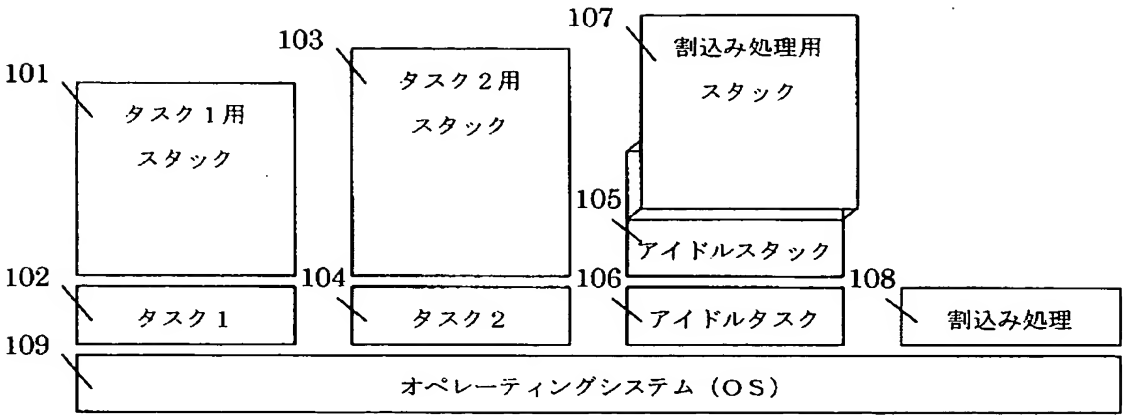
【００４５】

１０１　タスク１のスタック領域
１０２　タスク１のプログラム
１０３　タスク２のスタック領域
１０４　タスク２のプログラム
１０５　アイドルタスクのスタック領域
１０６　アイドルタスクのプログラム
１０７　割込み処理のスタック領域
１０８　割込み処理のプログラム

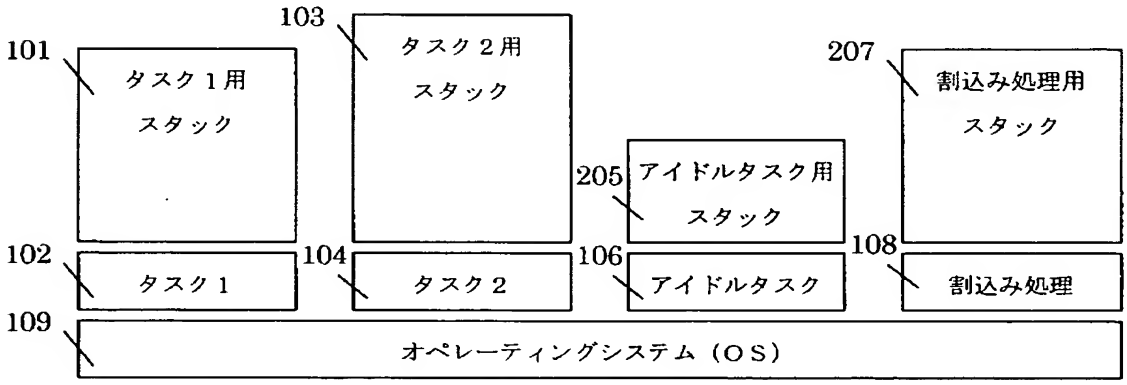
１０９　リアルタイムＯＳ
２０５　アイドルタスクのスタック領域（非破壊領域）
２０７　割込み処理のスタック領域
３０１　アイドルタスクが使用しているスタック領域
３０２　アイドルタスク動作中に割込みが発生した場合の戻りＰＣ保存領域
３０３　アイドルタスク動作中に割込みが発生した場合のＰＳＷ保存領域
３０４　アイドルタスク動作中に割込みが発生した場合のＣＰＵレジスタ保存領域
３０５　割込み発生時にそれまで動作していたタスクのＳＰレジスタ値を保存する領域
３０６　割込み処理で使用するスタック領域
３１１　タスク１が使用しているスタック領域
３１２　タスク１動作中に割込みが発生した場合の戻りＰＣ保存領域
３１３　タスク１動作中に割込みが発生した場合のＰＳＷ保存領域
３１４　タスク１動作中に割込みが発生した場合のＣＰＵレジスタ保存領域
３２１　タスク２が使用しているスタック領域
３２２　タスク２動作中に割込みが発生した場合の戻りＰＣ保存領域
３２３　タスク２動作中に割込みが発生した場合のＰＳＷ保存領域
３２４　タスク２動作中に割込みが発生した場合のＣＰＵレジスタ保存領域
４０１　アイドルタスクが使用しているスタック領域
４０２　アイドルタスク動作中に割込みが発生した場合の戻りＰＣ保存領域
４０３　アイドルタスク動作中に割込みが発生した場合のＰＳＷ保存領域
４０４　アイドルタスク動作中に割込みが発生した場合のＣＰＵレジスタ保存領域
４０５　割込み発生時にそれまで動作していたタスクのＳＰレジスタ値を保存する領域
４０６　割込み処理で使用するスタック領域
５０１　ＣＰＵレジスタをタスクスタックへ保存するステップ
５０２　タスク動作時のＳＰを割込みスタックのボトム位置へ退避するステップ
５０３　従来のＳＰの値を割込み処理用スタックへ切替えるステップ
５０４　アプリケーションの割込みハンドラを呼出すステップ
５０５　アプリケーションの割込みハンドラ処理を実行するステップ
６０３　実施の形態１におけるＳＰの値を割込み処理用スタックへ切替えるステップ
７０１　実施の形態２におけるＣＰＵレジスタをタスクスタックへ保存するステップ
７０３　実施の形態２におけるＳＰの値を割込み処理用スタックへ切替えるステップ
８０１　ＳＰの値をタスクスタックへ切替えるステップ
８０２　タスクスイッチが必要であれば遅延ディスパッチを行なうステップ
８０３　スタックに退避されているＣＰＵレジスタ値を復帰するステップ
８０４　割込み処理から復帰する命令を実行するステップ
９０１　自分自身への空ループを行なう処理ステップ
１００１　マイコンを低消費動作モードへ移行させるステップ
１１０１　アイドルタスクが使用しているスタック領域
１１０２　アイドルタスク動作中に割込みが発生した場合の戻りＰＣ保存領域
１１０３　アイドルタスク動作中に割込みが発生した場合のＰＳＷ保存領域
１１０４　アイドルタスク動作中に割込みが発生した場合のＣＰＵレジスタ保存領域
１１０５　割込み発生時にそれまで動作していたタスクのＳＰレジスタ値を保存する領域
１１０６　割込み処理で使用するスタック領域
１１０７　ｒ０レジスタ保存領域
１２００　実施の形態１および２におけるタスク制御装置
１２０１　汎用ＣＰＵレジスタ
１２０２　スタックポインタ（ＳＰ）

１２０３　プロセッサステータスワード（ＰＳＷ）
１２０４　演算ユニット（ＡＬＵ）
１２０５　プログラムカウンタ（ＰＣ）
１２０６　命令実行制御部（命令デコーダ）
１２０７　オペランドアドレス
１２０８　オペランドデータ
１２０９　命令アドレス
１２１０　命令データ
１２１１　割込み制御部
１２１２　バスコトローラユニット（ＢＣＵ）
１２１３　読込み専用外部メモリ（ＲＯＭ）
１２１４　読書き可能外部メモリ（ＲＡＭ）
１２１５　周辺機能
１２５０　ＳＰ差分定数
１３００　ＳＰ差分情報格納レジスタ

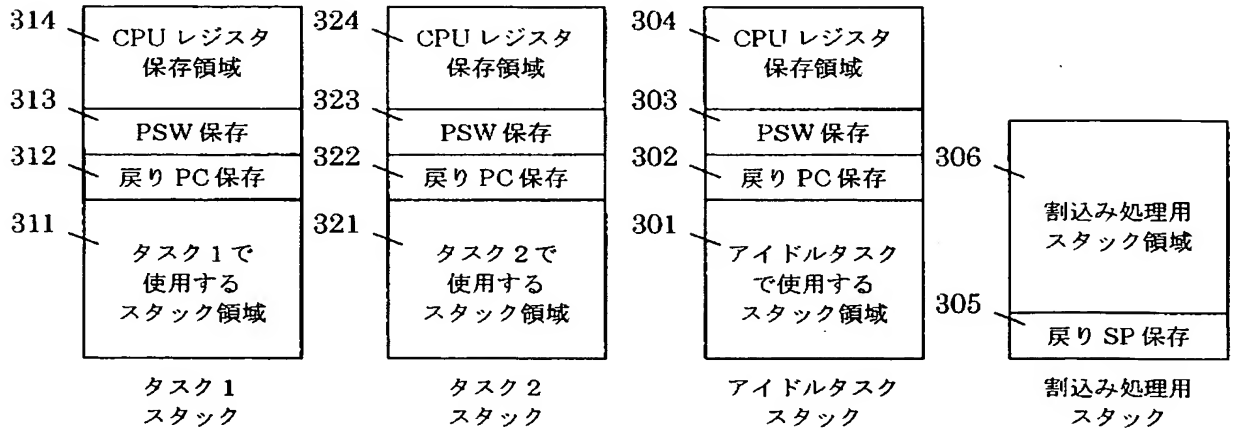
【書類名】図面
【図１】



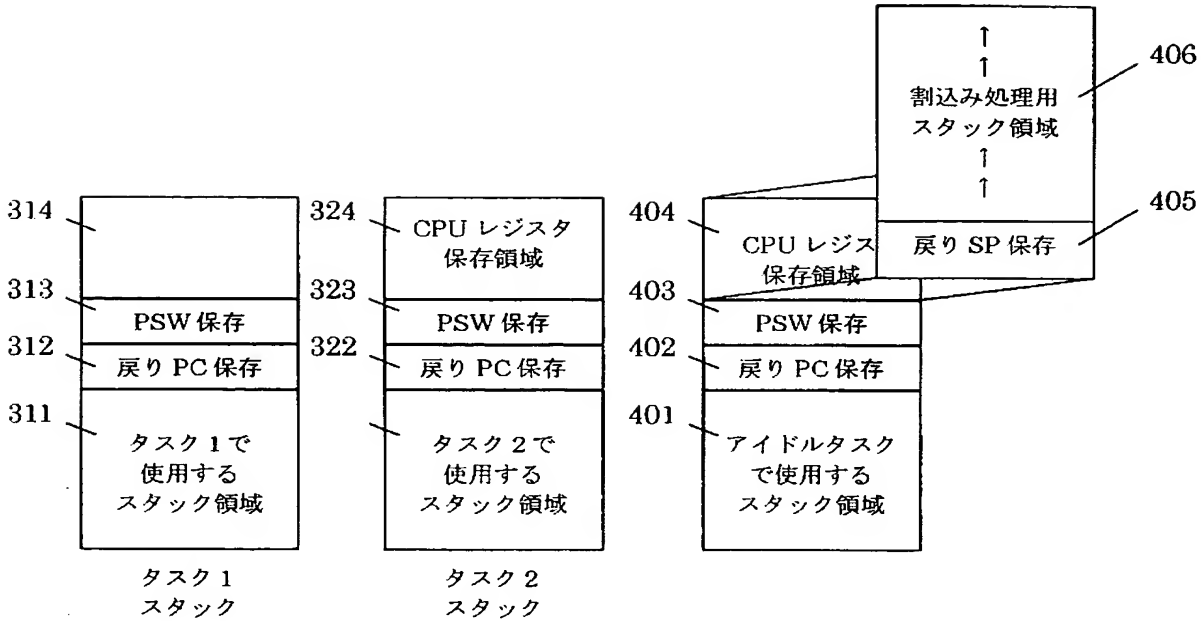
【図２】



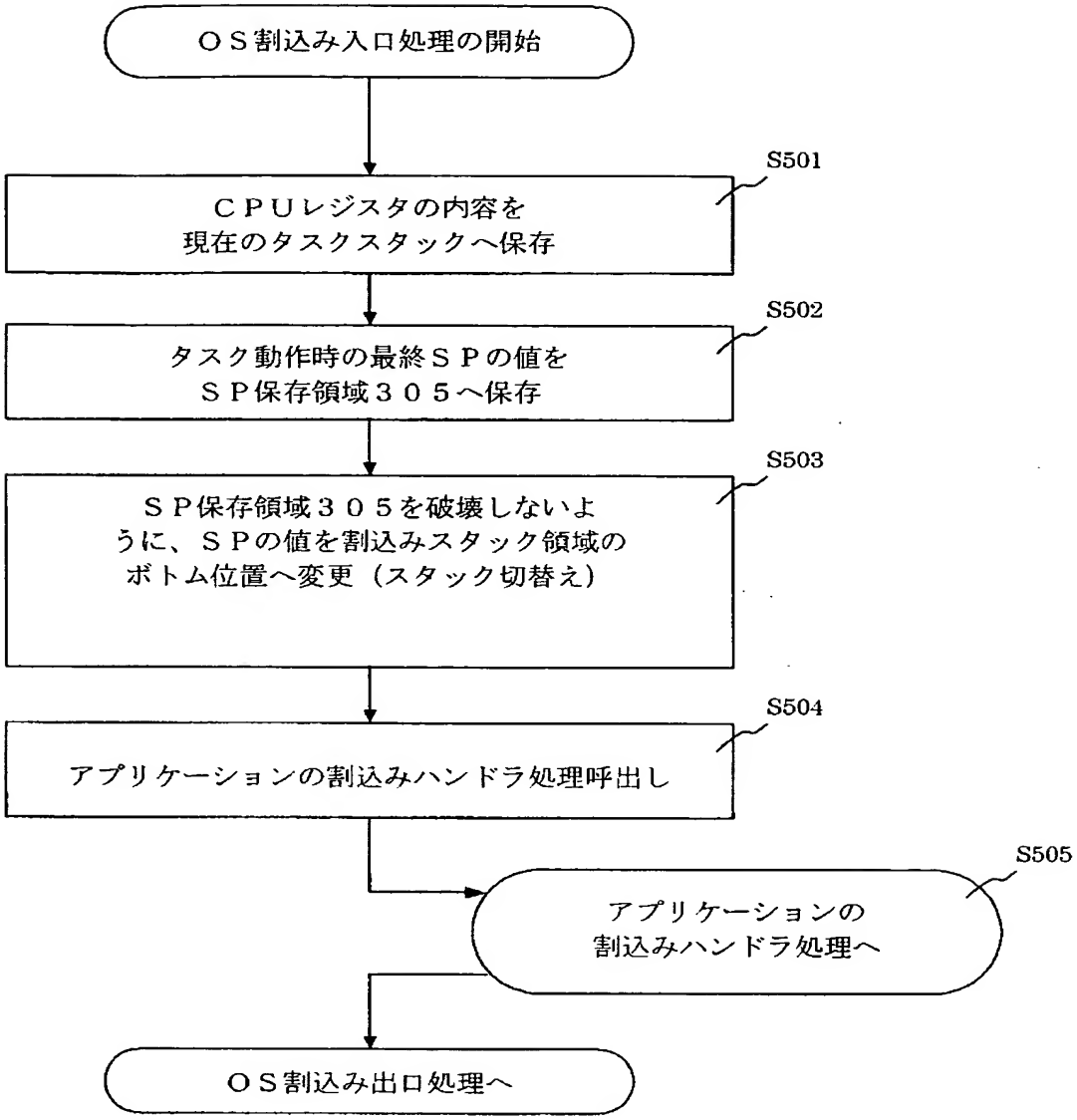
【図３】



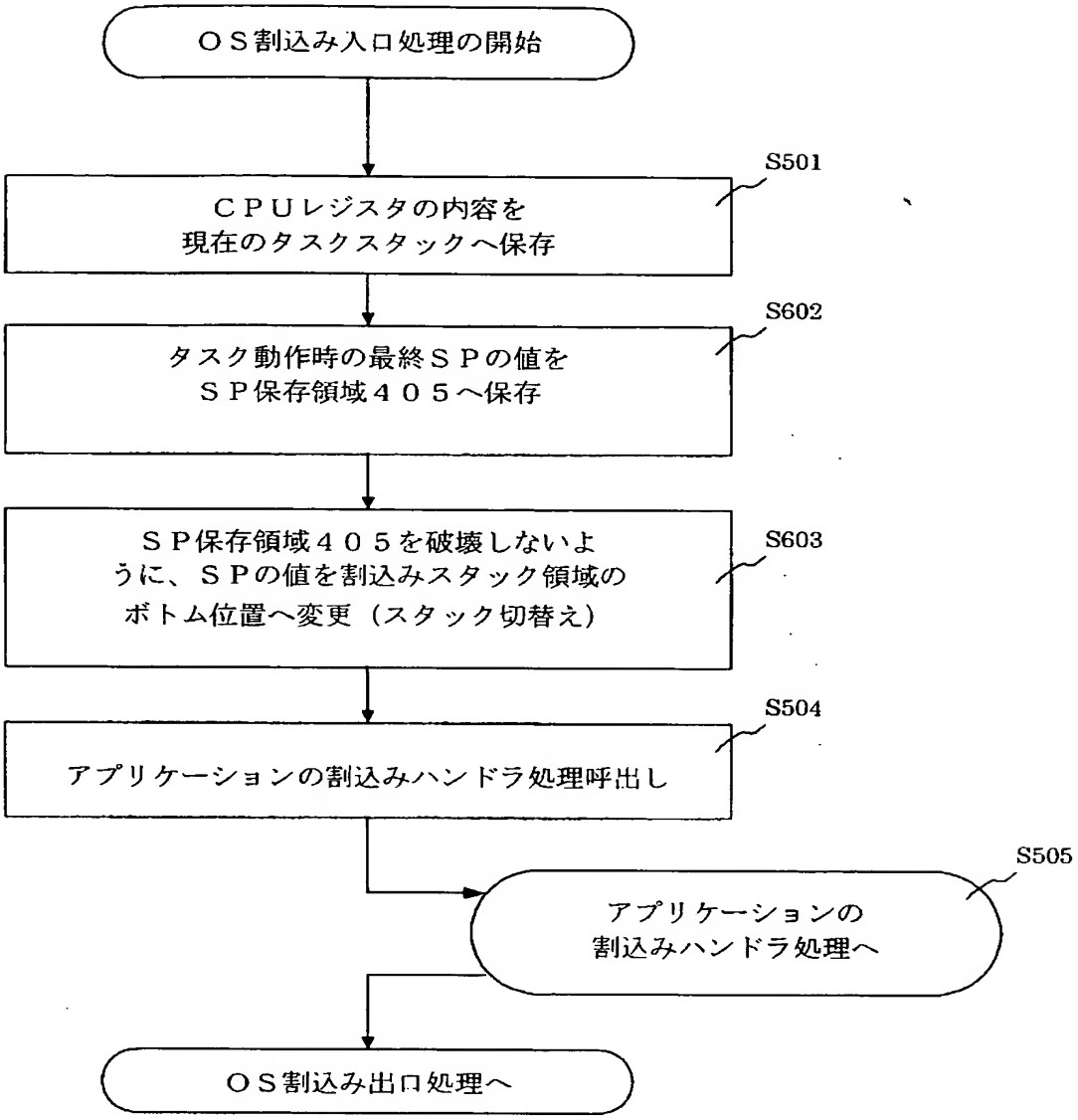
【図４】


406
割込み処理用
スタック領域
405
戻り SP 保存
314
313
PSW 保存
312
戻り PC 保存
311
タスク１で
使用する
スタック領域
タスク１
スタック
324
CPU レジスタ
保存領域
323
PSW 保存
322
戻り PC 保存
タスク２で
使用する
スタック領域
タスク２
スタック
404
CPU レジス
保存領域
403
PSW 保存
402
戻り PC 保存
401
アイドルタスク
で使用する
スタック領域

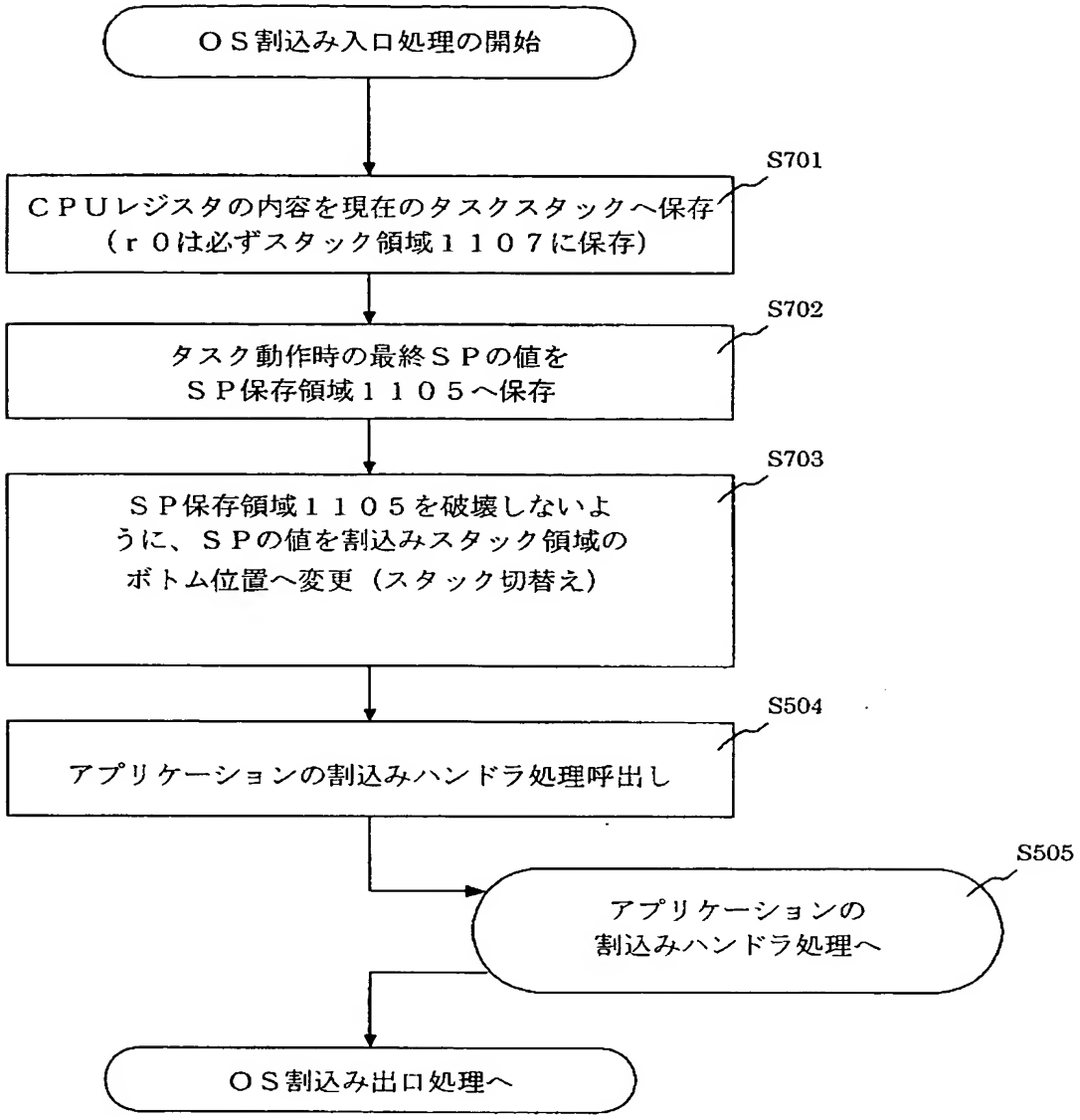
【図５】


ＯＳ割込み入口処理の開始
S501
ＣＰＵレジスタの内容を
現在のタスクスタックへ保存
S502
タスク動作時の最終ＳＰの値を
ＳＰ保存領域３０５へ保存
S503
ＳＰ保存領域３０５を破壊しないよ
うに、ＳＰの値を割込みスタック領域の
ボトム位置へ変更（スタック切替え）
S504
アプリケーションの割込みハンドラ処理呼出し
S505
アプリケーションの
割込みハンドラ処理へ
ＯＳ割込み出口処理へ

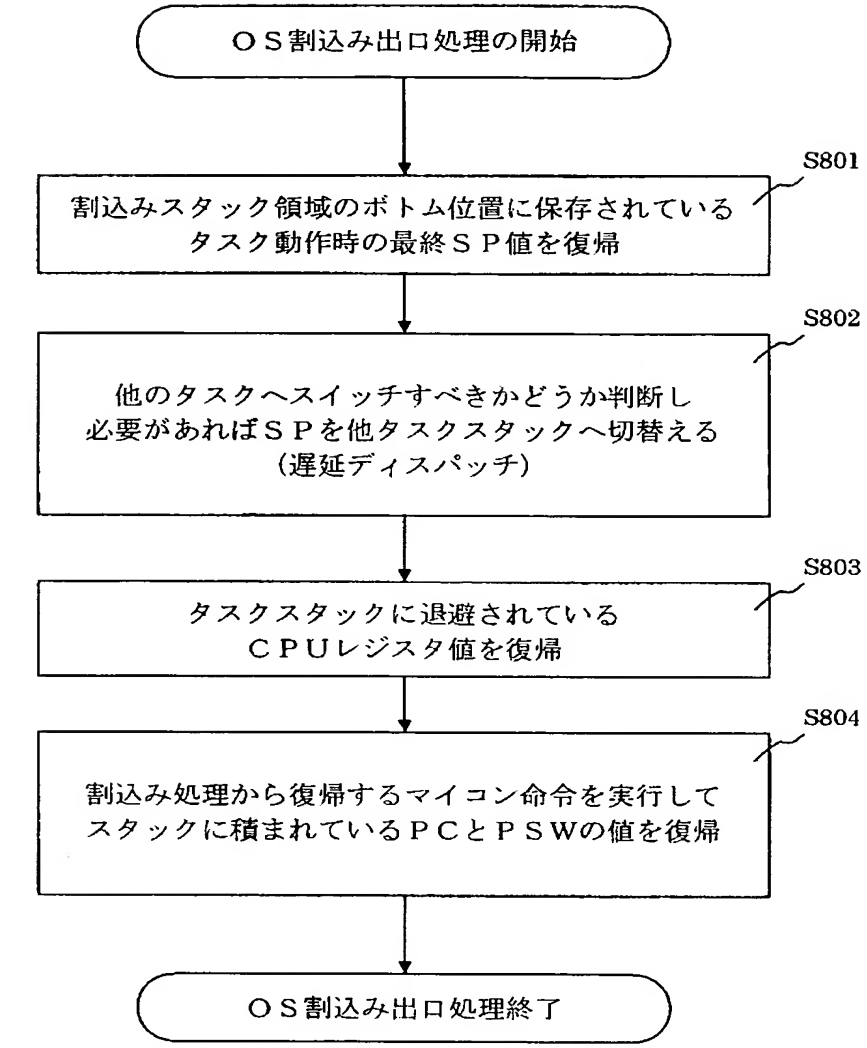
【図６】


ＯＳ割込み入口処理の開始
S501
ＣＰＵレジスタの内容を
現在のタスクスタックへ保存
S602
タスク動作時の最終ＳＰの値を
ＳＰ保存領域４０５へ保存
S603
ＳＰ保存領域４０５を破壊しないよ
うに、ＳＰの値を割込みスタック領域の
ボトム位置へ変更（スタック切替え）
S504
アプリケーションの割込みハンドラ処理呼出し
S505
アプリケーションの
割込みハンドラ処理へ
ＯＳ割込み出口処理へ

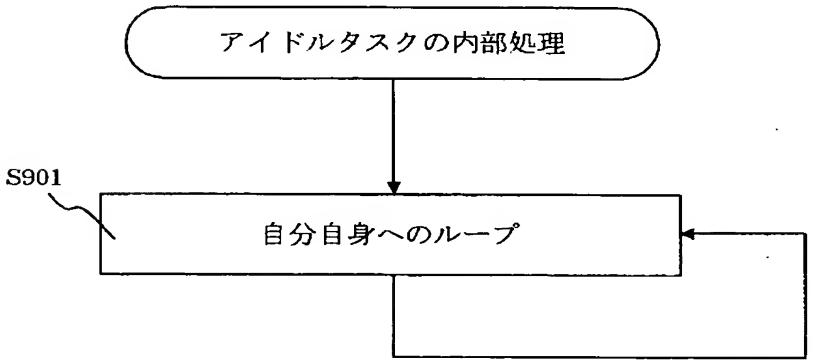
【図７】


ＯＳ割込み入口処理の開始
S701
ＣＰＵレジスタの内容を現在のタスクスタックへ保存
（ｒ０は必ずスタック領域１１０７に保存）
S702
タスク動作時の最終ＳＰの値を
ＳＰ保存領域１１０５へ保存
S703
ＳＰ保存領域１１０５を破壊しないよ
うに、ＳＰの値を割込みスタック領域の
ボトム位置へ変更（スタック切替え）
S504
アプリケーションの割込みハンドラ処理呼出し
S505
アプリケーションの
割込みハンドラ処理へ
ＯＳ割込み出口処理へ

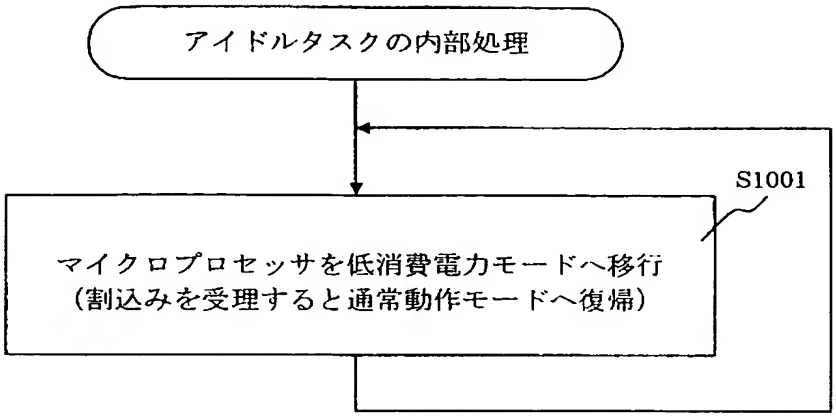
【図８】


ＯＳ割込み出口処理の開始
S801
割込みスタック領域のボトム位置に保存されている
タスク動作時の最終ＳＰ値を復帰
S802
他のタスクへスイッチすべきかどうか判断し
必要があればＳＰを他タスクスタックへ切替える
（遅延ディスパッチ）
S803
タスクスタックに退避されている
ＣＰＵレジスタ値を復帰
S804
割込み処理から復帰するマイコン命令を実行して
スタックに積まれているＰＣとＰＳＷの値を復帰
ＯＳ割込み出口処理終了

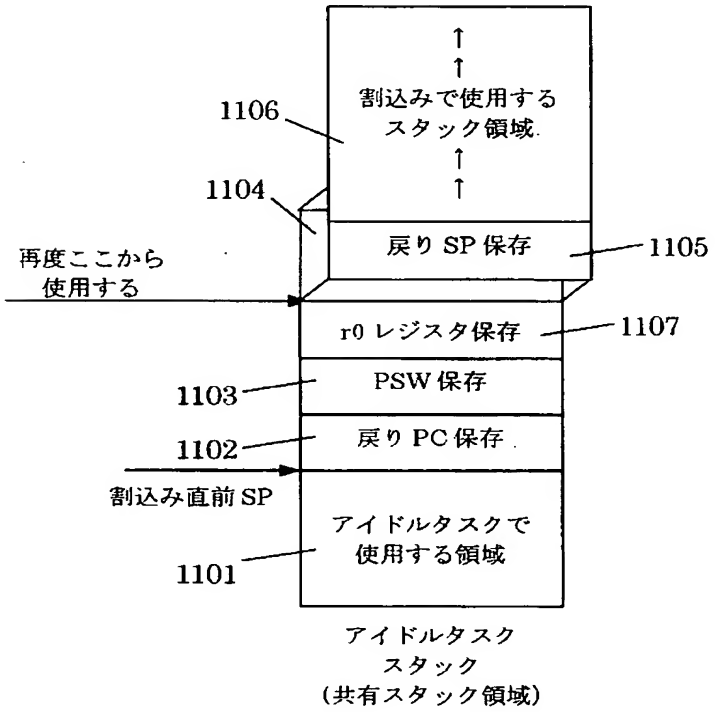
【図９】


アイドルタスクの内部処理
S901
自分自身へのループ

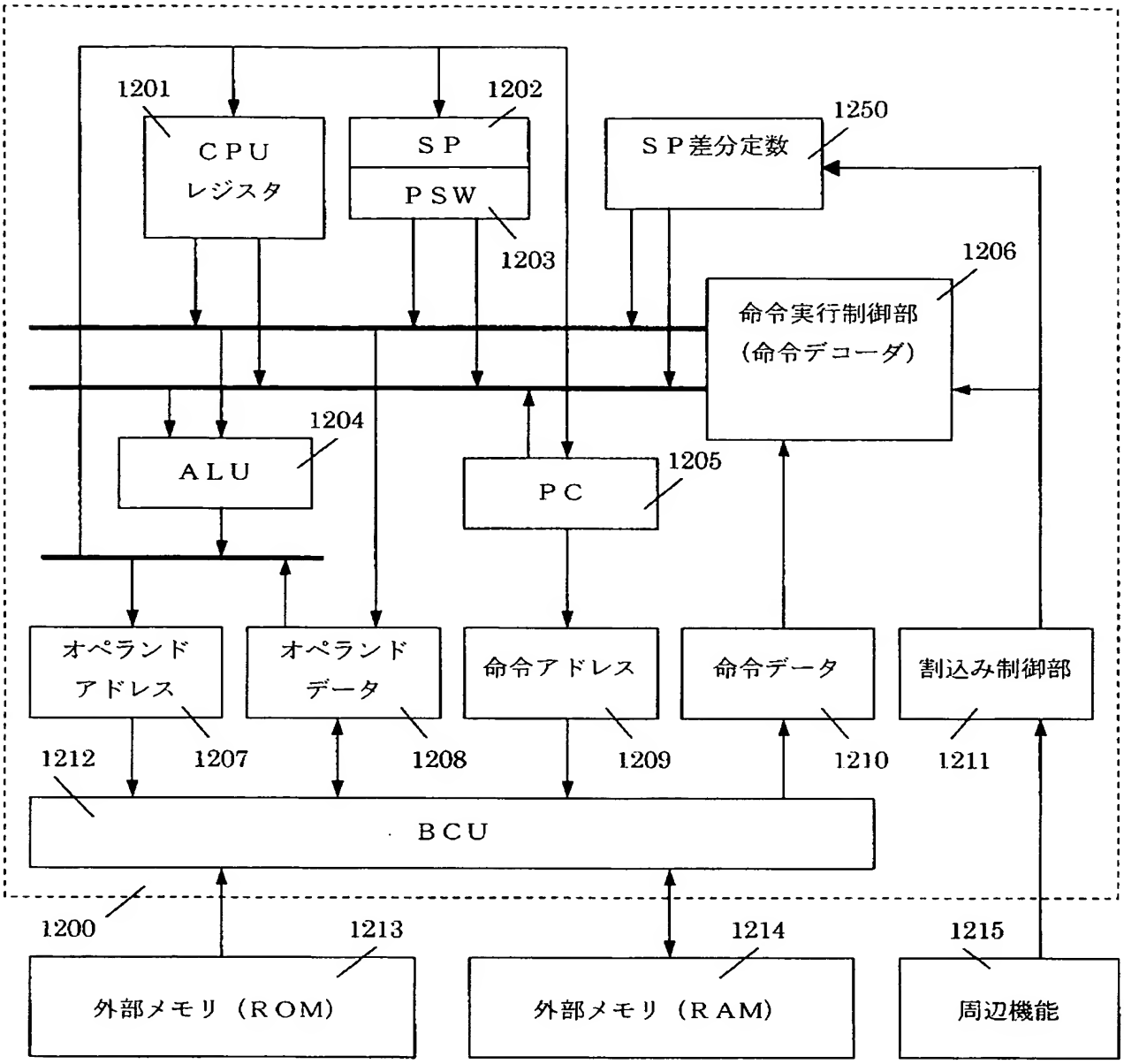
【図１０】


アイドルタスクの内部処理
S1001
マイクロプロセッサを低消費電力モードへ移行
（割込みを受理すると通常動作モードへ復帰）

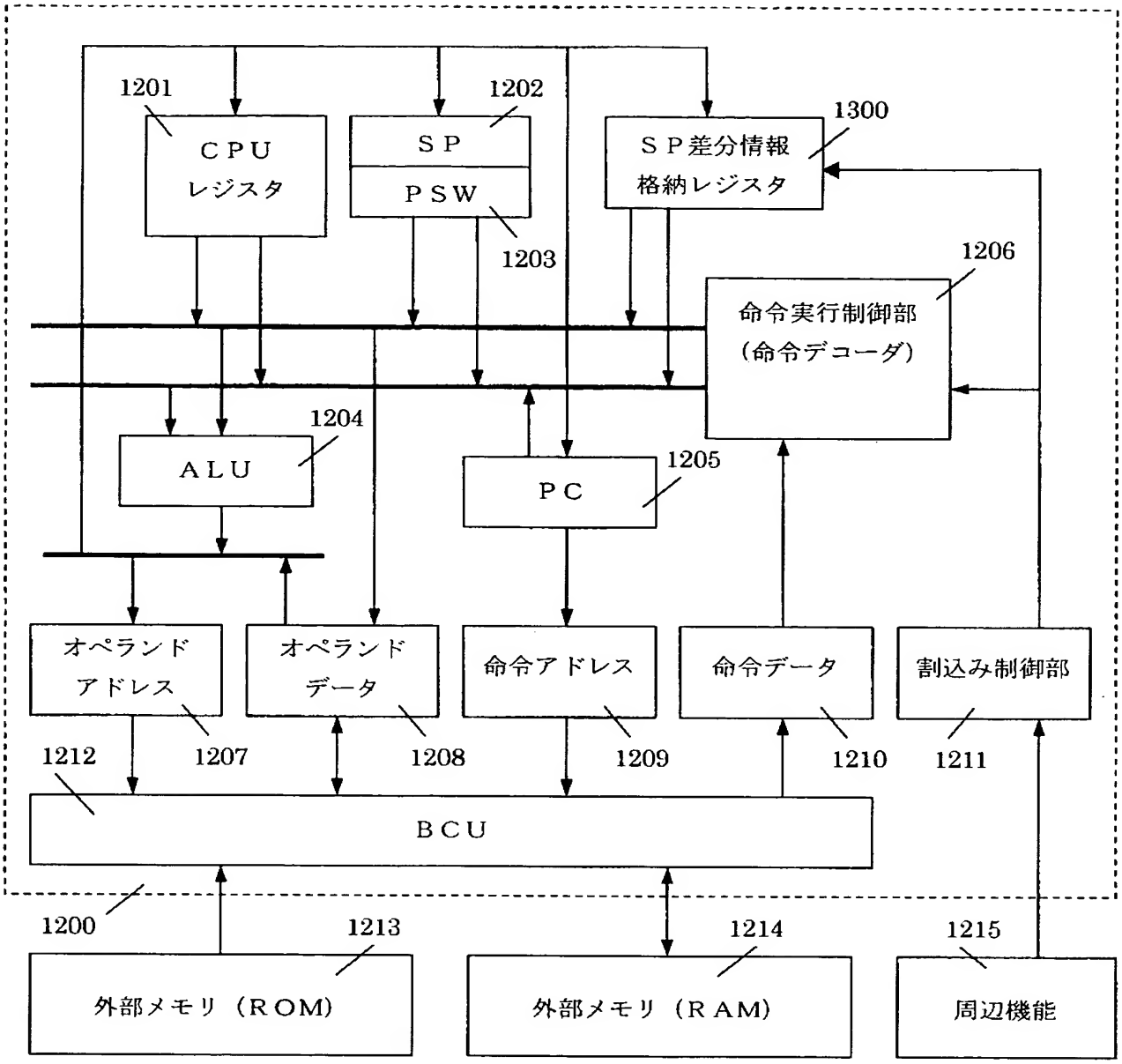
【図１１】


割込みで使用する
スタック領域
1106
1104
戻り SP 保存
1105
再度ここから
使用する
r0 レジスタ保存
1107
PSW 保存
1103
戻り PC 保存
1102
割込み直前 SP
アイドルタスクで
使用する領域
1101
アイドルタスク
スタック
（共有スタック領域）

【図１２】


1201
CPU
レジスタ
1202
SP
PSW
1203
1250
SP差分定数
1206
命令実行制御部
(命令デコーダ)
1204
ALU
1205
PC
オペランド
アドレス
オペランド
データ
命令アドレス
命令データ
割込み制御部
1212
1207
1208
1209
1210
1211
BCU
1200
1213
1214
1215
外部メモリ（ROM）
外部メモリ（RAM）
周辺機能

【図１３】


1201
CPU
レジスタ
1202
SP
PSW
1203
1300
SP差分情報
格納レジスタ
1206
命令実行制御部
(命令デコーダ)
1204
ALU
1205
PC
オペランド
アドレス
オペランド
データ
命令アドレス
命令データ
割込み制御部
1212
1207
1208
1209
1210
1211
BCU
1200
1213
1214
1215
外部メモリ（ROM）
外部メモリ（RAM）
周辺機能

【書類名】要約書
【課題】マルチタスクシステムにおいて、割込み処理が使用するスタックをアイドル処理が使用するスタックと共有することで、ＲＡＭ使用量を削減する。
【解決手段】アイドルタスク動作中に割込みが発生すると、現在のスタック領域１０５にＣＰＵレジスタの値を保存した後、割込み処理専用のスタック領域１０７に切替える。このとき、スタック領域１０５と、割込み処理専用のスタック領域１０７が重なるようなスタックの構成にしておく。アイドル処理中に割込みが発生した場合は、ＣＰＵレジスタの値を保存した領域を上書きするように割込み処理のスタックを使用する。
【選択図】　図１

特願２００３－２８２８８７

出　願　人　履　歴　情　報

識別番号　　［０００００５８２１］

１．変更年月日　　１９９０年　８月２８日
［変更理由］　　新規登録
住　所　　大阪府門真市大字門真１００６番地
氏　名　　松下電器産業株式会社